

(19)



JAPANESE PATENT OFFICE

PATENT ABSTRACTS OF JAPAN

(11) Publication number: 2001318785 A

(43) Date of publication of application: 16.11.01

(51) Int. Cl.

G06F 7/72
G09C 1/00

(21) Application number: 2000137182

(22) Date of filing: 10.05.00

(71) Applicant: TOSHIBA CORP

(72) Inventor: KOIKE MASANOBU
KAWAMURA SHINICHI

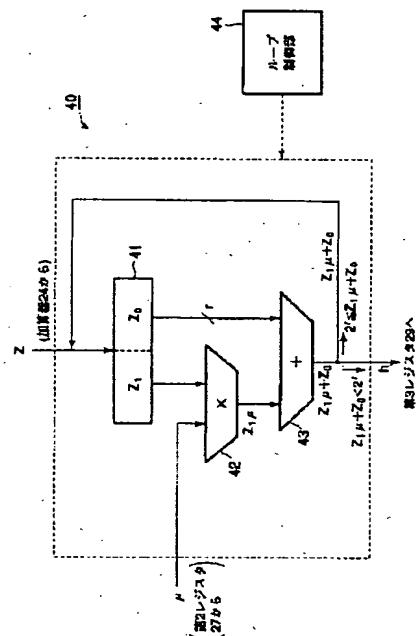
(54) MONTGOMERY MULTIPLICATION DEVICE AND METHOD

(57) Abstract:

PROBLEM TO BE SOLVED: To accelerate at least partial remainder operations and to accelerate the overall operation speed further by the acceleration of remainder computation.

SOLUTION: In a division circuit 40 arranged in respective product sum circuits 301-30n, though it is the remainder computation by a modulus m , the end condition of the remainder operation is mitigated so as to compare the executed result h' of the remainder computation with a value $2r$ larger than the modulus m . Thus, the remainder computation is accelerated by the acceleration of loop control itself by the adoption of an easily comparable number which is $2r$ and the reduction of the number of times of loops by the adoption of a large number which is $2r$.

COPYRIGHT: (C)2001,JPO



JP-A-2001-318785

[What is claimed is:]

1. A Montgomery multiplication device
 for, when a positive integer N and positive integers
 5 x and y that are less than N are set to inputs and bases
 of a remainder computation system are set to $\{a_1, a_2, \dots,$
 $a_i, \dots, a_n\}$ and $\{b_1, b_2, \dots, b_i, \dots, b_n\}$ ($1 \leq i \leq n$, n is a positive
 integer), calculating an output w that is equal to xyB^{-1}
 mod N using an integer B that is defined as multiplication
 10 of elements of the bases, comprising:
 - a first storage unit storing elements of the
 respective bases $\{a_1, a_2, \dots, a_n\}$ and $\{b_1, b_2, \dots, b_n\}$ and
 a prior calculation result used for calculating the
 output w ;
 - 15 an approximation computation unit, when an input
 ξ_i used for expansion of the bases is received, extracting
 higher q bits of the input ξ_i , adding a present extraction
 value and a previous addition result excluding a highest
 bit and outputting a value k_i of a highest bit from the
 20 obtained addition result;
 - a remainder computation unit executing remainder
 calculation (modulus m that is equal to $2^r - \mu$ is one element
 of the bases, r is an integer that satisfies $m \leq 2^r$ and
 μ is a nonnegative integer that is less than 2^r) while
 25 setting elements of the respective bases $\{a_1, a_2, \dots, a_n\}$

and $\{b_1, b_2, \dots, b_n\}$ to a modulus m based on contents of the first storage unit and the output k_1 of the approximation computation unit; and

a second storage unit storing results of remainder calculation performed by the remainder computation unit, wherein

the remainder computation unit comprises:

a remainder calculation part for receiving an input z (z is a positive integer) comprising a sum $(xy+d+c)$ of a product xy of inputs x and y inputted from the first or second storage unit, a previous remainder calculation result d and the prior calculation result c inputted based on the output k_1 of the approximate computation unit, and for executing remainder calculation $h = z \bmod m$ by a modulus m of the input z ; and

a loop control part for comparing an execution result h' executed by the remainder calculation part with an upper limit value 2^r of the modulus m , for returning the execution result h' to the remainder calculation part as the input z when the execution result h' is equal to or greater than 2^r and for outputting the execution result h' to the second storing unit as a result h of the remainder calculation when the execution result h' is less than 2^r .

2. The Montgomery multiplication device according to claim 1, wherein the remainder calculation part, comprises:

5 a bit selection part for, when executing remainder calculation $h = z \bmod m$ by a modulus m of the input z , respectively selecting a value z_0 of lower r bits of the input z and a value z_1 of higher bits excluding
10 the value z_0 of the lower bits from the input z ;

a multiplication part for, when the value z_1 of higher bits extracted by the bit selection part and an input μ regarding the modulus m are received,
15 calculating a product $z_1\mu$ of the value z_1 and the input μ ; and

an addition part for calculating a sum $(z_1\mu + z_0)$ of the product $z_1\mu$ obtained by the multiplication part and the value z_0 of lower r bits extracted by the
20 bit selection part, wherein

an execution result h' executed by the remainder calculation part is a sum $(z_1\mu + z_0)$ obtained by the addition part.

25 3. A Montgomery multiplication method

of, when a positive integer N and positive integers x and y that are less than N are set to inputs and bases of a remainder computation system are set to $\{a_1, a_2, \dots, a_i, \dots, a_n\}$ and $\{b_1, b_2, \dots, b_i, \dots, b_n\}$ ($1 \leq i \leq n$, n is a positive integer), calculating an output w that is equal to $xyB^{-1} \bmod N$ using an integer B that is defined as multiplication of elements of the bases, comprising:

a first storage process of storing elements of the respective bases $\{a_1, a_2, \dots, a_n\}$ and $\{b_1, b_2, \dots, b_n\}$ and a prior calculation result used for calculating the output w ;

an approximation computation process of, when an input ξ_i used for expansion of the bases is received, extracting higher q bits of the input ξ_i , adding a present extraction value and a previous addition result excluding a highest bit and outputting a value k_i of a highest bit from the obtained addition result;

a remainder computation process of executing remainder calculation³ (modulus m that is equal to $2^r - \mu$ is one element of the bases, r is an integer that satisfies $m \leq 2^r$ and μ is a nonnegative integer that is less than 2^r) while elements of the respective bases $\{a_1, a_2, \dots, a_n\}$ and $\{b_1, b_2, \dots, b_n\}$ are set to a modulus m based on contents of the first storage process and the output k_i of the approximation computation process; and

a second storage process of storing results of remainder calculation performed by the remainder computation process, wherein

the remainder computation process comprises:

5 a remainder calculation process of receiving an input z (z is a positive integer) comprising a sum ($xy+d+c$) of a product xy of inputs x and y inputted from storage contents of the first or second storage process, a previous remainder calculation result d and the prior
10 calculation result c inputted based on the output k_i of the approximate computation process, and of executing remainder calculation $h = z \bmod m$ by a modulus m of the input z ; and

a loop control process of comparing an execution
15 result h' executed by the remainder calculation process with an upper limit value 2^r of the modulus m , returning the execution result h' to the remainder calculation process as the input z when the execution result h' is equal to or greater than 2^r and outputting the execution
20 result h' to the second storing process as a result h of the remainder calculation when the execution result h' is less than 2^r .

4. The Montgomery multiplication method
25 according to claim 3, wherein

the remainder calculation process,
comprises:

- a bit selection process of, when
executing remainder calculation $h = z \bmod m$ by a modulus
5 m of the input z , respectively selecting a value z_0 of
lower r bits of the input z and a value z_1 of higher bits
excluding
the value z_0 of the lower bits from the input
 z ;
- 10 a multiplication process of, when the
value z_1 of higher bits extracted by the bit selection
process and an input μ regarding the modulus m are received,
calculating a product $z_1\mu$ of the value z_1 and the input
 μ ; and
- 15 an addition process of calculating a
sum $(z_1\mu + z_0)$ of the product $z_1\mu$ obtained by the
multiplication process and the value z_0 of the lower r
bits extracted by the bit selection process, wherein
an execution result h' executed by the remainder
20 calculation process is a sum $(z_1\mu + z_0)$ obtained by the
addition process.

(19)日本国特許庁 (J P)

(12) 公 開 特 許 公 報 (A)

(11)特許出願公開番号
特開2001-318785
(P2001-318785A)

(43)公開日 平成13年11月16日 (2001.11.16)

(51)Int.Cl. ⁷	識別記号	F I	ターミナル (参考)
G 0 6 F 7/72		G 0 6 F 7/72	5 J 1 0 4
G 0 9 C 1/00	6 5 0	G 0 9 C 1/00	6 5 0 A 9 A 0 0 1

審査請求 未請求 請求項の数4 O L (全 12 頁)

(21)出願番号 特願2000-137182(P2000-137182)

(22)出願日 平成12年5月10日(2000.5.10)

(71)出願人 000003078

株式会社東芝

東京都港区芝浦一丁目1番1号

(72)発明者 小池 正修

東京都府中市東芝町1番地 株式会社東芝
府中工場内

(72)発明者 川村 信一

神奈川県川崎市幸区小向東芝町1番地 株
式会社東芝研究開発センター内

(74)代理人 100058479

弁理士 鈴江 武彦 (外6名)

Fターム(参考) 5J104 AA22 JA23 NA18

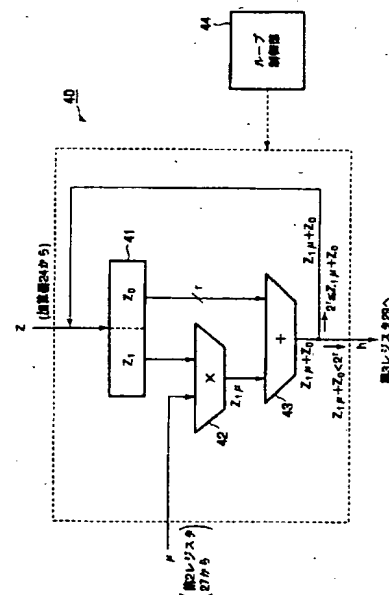
9A001 BB02 EE03 GG01

(54)【発明の名称】 モンゴメリ乗算装置及び方法

(57)【要約】

【課題】 剰余算の高速化により、少なくとも部分的な剰余演算を高速化でき、さらに全体的な演算速度の高速化を実現させる。

【解決手段】 各積和回路30₁~30_nに配置された除算回路40において、法mでの剰余算であるにも関わらず、剰余算の実行結果h'を法mよりも大きい値2^rと比較するように剰余算の終了条件を緩和している。これにより、2^rという比較の容易な数の採用によるループ制御自体の高速化と、2^rという大きい数の採用によるループ回数の低減化とによって剰余算を高速化させる。



【特許請求の範囲】

【請求項1】 正の整数 N と、 N 未満の正の整数 x 、 y とを入力とし、剰余演算系の基底を $\{a_1, a_2, \dots, a_i, \dots, a_n\}$ および $\{b_1, b_2, \dots, b_i, \dots, b_n\}$ ($1 \leq i \leq n$ 、 n は正の整数)としたとき、前記基底の要素の乗積として定義された整数 $B (= b_1 b_2 \dots b_n)$ を用いて、出力 $w = xyB^{-1} \bmod N$ を算出するためのモンゴメリ乗算装置であって、

前記各基底の要素 $\{a_1, a_2, \dots, a_n\}$ および $\{b_1, b_2, \dots, b_n\}$ と前記出力 w の算出用の事前計算結果とが記憶される第1記憶手段と、

前記基底の拡張用の入力 ξ_i を受けたとき、前記入力 ξ_i の上位 q ビットを抽出すると共に、この今回の抽出値と前回の最上位ビットを除く加算結果とを加算し、得られた加算結果のうち、最上位ビットの値 k_i を出力する近似演算手段と、

前記第1記憶手段の内容および前記近似演算手段の出力 k_i に基づいて、前記各基底の要素 $\{a_1, a_2, \dots, a_n\}$ および $\{b_1, b_2, \dots, b_n\}$ を法 m とする剰余算(但し、法 $m = 2^r - \mu$ は基底の一要素、 r は $m \leq 2^r$ を満たす整数、 μ は 2^r 未満の非負整数)を実行する剰余演算手段と、

前記剰余演算手段による剰余算の結果が記憶される第2記憶手段とを備え、

前記剰余演算手段は、

前記第1記憶手段又は前記第2記憶手段からの入力 x 、 y の積 xy と前回の剰余算の結果 d と前記近似演算手段の出力 k_i に基づき入力される前記事前計算結果 c との和 $(xy + d + c)$ からなる入力 z (但し、 z は正の整数)を受けると共に、前記入力 z の法 m による剰余算 $h = z \bmod m$ を実行する剰余算部と、

前記剰余算部による実行結果 h' と前記法 m の上限値 2^r とを比較し、前記実行結果 h' が 2^r 以上のときには当該実行結果 h' を前記入力 z として前記剰余算部に戻し、前記実行結果 h' が 2^r 未満のときには当該実行結果 h' を前記剰余算の結果 h として前記第2記憶手段に出力するループ制御部と、

を備えたことを特徴とするモンゴメリ乗算装置。

【請求項2】 請求項1に記載のモンゴメリ乗算装置において、

前記剰余算部は、

前記入力 z の法 m による剰余算 $h = z \bmod m$ を実行するとき、前記入力 z の下位 r ビットの値 z_0 と、前記入力 z から前記下位の値 z_0 を除いた上位ビットの値 z_1 とを夫々抽出するビット選択部と、

前記ビット選択部により抽出された上位ビットの値 z_1 と前記法 m に関する入力 μ とを受けたとき、両者の積 $z_1 \mu$ を算出する乗算部と、

前記乗算部により得られた積 $z_1 \mu$ と前記ビット選択部により抽出された下位 r ビットの値 z_0 との和 $(z_1 \mu$

$+ z_0)$ を算出する加算部とを備え、

前記剰余算部による実行結果 h' は、前記加算部により得られた和 $(z_1 \mu + z_0)$ であることを特徴とするモンゴメリ乗算装置。

【請求項3】 正の整数 N と、 N 未満の正の整数 x 、 y とを入力とし、剰余演算系の基底を $\{a_1, a_2, \dots, a_i, \dots, a_n\}$ および $\{b_1, b_2, \dots, b_i, \dots, b_n\}$ ($1 \leq i \leq n$ 、 n は正の整数)としたとき、前記基底の要素の乗積として定義された整数 $B (= b_1 b_2 \dots b_n)$ を用いて、出力 $w = xyB^{-1} \bmod N$ を算出するためのモンゴメリ乗算方法であって、

前記各基底の要素 $\{a_1, a_2, \dots, a_n\}$ および $\{b_1, b_2, \dots, b_n\}$ と前記出力 w の算出用の事前計算結果とが記憶される第1記憶工程と、

前記基底の拡張用の入力 ξ_i を受けたとき、前記入力 ξ_i の上位 q ビットを抽出すると共に、この今回の抽出値と前回の最上位ビットを除く加算結果とを加算し、得られた加算結果のうち、最上位ビットの値 k_i を出力する近似演算工程と、

20 前記第1記憶工程の記憶内容および前記近似演算工程の出力 k_i に基づいて、前記各基底の要素 $\{a_1, a_2, \dots, a_n\}$ および $\{b_1, b_2, \dots, b_n\}$ を法 m とする剰余算(但し、法 $m = 2^r - \mu$ は基底の一要素、 r は $m \leq 2^r$ を満たす整数、 μ は 2^r 未満の非負整数)を実行する剰余演算工程と、

前記剰余演算工程による剰余算の結果が記憶される第2記憶工程とを含んでおり、

前記剰余演算工程は、

30 前記第1記憶工程の記憶内容又は前記第2記憶工程の記憶内容からの入力 x 、 y の積 xy と前回の剰余算の結果 d と前記近似演算工程の出力 k_i に基づき入力される前記事前計算結果 c との和 $(xy + d + c)$ からなる入力 z (但し、 z は正の整数)を受けると共に、前記入力 z の法 m による剰余算 $h = z \bmod m$ を実行する剰余算工程と、

前記剰余算工程による実行結果 h' と前記法 m の上限値 2^r とを比較し、前記実行結果 h' が 2^r 以上のときには当該実行結果 h' を前記入力 z として前記剰余算工程に戻し、前記実行結果 h' が 2^r 未満のときには当該実行結果 h' を前記剰余算の結果 h として前記第2記憶工程に出力するループ制御工程と、

を含んでいることを特徴とするモンゴメリ乗算方法。

【請求項4】 請求項3に記載のモンゴメリ乗算方法において、

前記剰余算工程は、

前記入力 z の法 m による剰余算 $h = z \bmod m$ を実行するとき、前記入力 z の下位 r ビットの値 z_0 と、前記入力 z から前記下位の値 z_0 を除いた上位ビットの値 z_1 とを夫々抽出するビット選択工程と、

50 前記ビット選択工程により抽出された上位ビットの値 z

1と前記法mに関する入力 μ とを受けたとき、両者の積 $z_1\mu$ を算出する乗算工程と、前記乗算工程により得られた積 $z_1\mu$ と前記ビット選択工程により抽出された下位 r ビットの値 z_0 との和 $(z_1\mu + z_0)$ を算出する加算工程とを含んでおり、前記剰余算工程による実行結果 h' は、前記加算工程により得られた和 $(z_1\mu + z_0)$ であることを特徴とするモンゴメリ乗算方法。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】本発明は、剰余演算系におけるモンゴメリ乗算アルゴリズムを用いるモンゴメリ乗算装置及び方法に係り、特に、剰余算のループ制御の高速化により、全体の処理速度を向上し得るモンゴメリ乗算装置及び方法に関する。

【0002】

【従来の技術】近年、公開鍵暗号の重要性が増すにつれ、公開鍵暗号における大きな整数の剰余乗算に関し、その高速化を図る旨の要望が高まっている。例えば現在、事実上の標準（デファクトスタンダード）であるR

SA (Rivest-Shamir-Adleman) 方式では、1,024ビットの整数による剰余演算を高速に行なう必要がある。

【0003】係る剰余演算の高速化の一方法として、剰余演算系 (Residue Number System、以後RNSという) が知られている。このRNSは、互いに素な比較的小さな整数の組 $\{a_1, a_2, \dots, a_n\}$ を用意し、演算対象の大きな整数をこれらの整数で割った余りの組で表現する方式である。

【0004】以後、これら割る数の組 $\{a_1, a_2, \dots, a_n\}$ をRNSの基底と呼び、基底を構成する要素

の個数 n を基底のサイズと呼ぶ。また基底の集合を $a = \{a_1, a_2, \dots, a_n\}$ で表す。

【0005】例えば、整数 x と基底 $\{a_1, a_2, \dots, a_n\}$ が与えられたとき、 x を基底 a_i ($i=1, 2, \dots, n$) で割った余り x_i の組 (x_1, x_2, \dots, x_n) が x のRNS表現である。ここで、 $x_i = x \bmod a_i$ である。

【0006】このとき、 x は、基底の全要素の積 $A = a_1 a_2 \dots a_n$ を法として一意的に表現できる。すなわち、 x が A 未満の正整数であれば、 x とそのRNS表現 (x_1, x_2, \dots, x_n) は一対一に対応する。

【0007】RNS表現の数値同士の剰余乗算方法は、ポッシュら (Posch et al) による提案 (K. C. Posch, R. Posch, "Modulo Reduction in Residue Number Systems", IEEE Transaction on Parallel and Distributed Systems, Vol.6, No.5, May 1995, pp.449-454 及び "RNS-Modulo Reduction Upon a Restricted Base Value Set and its Applicability to RSA Cryptography", Computer & Security, Vol.17, pp.637-650, 1998) 並びに川村による提案 (特願平11-310619

号) がある。

【0008】これらの剰余乗算方法は、RNS表現での演算において不利な除算を避けるために、モンゴメリ (Montgomery) の提案による剰余乗算方法 (以下、モンゴメリ乗算という。詳細は P. L. Montgomery, "Modular Multiplication without Trial Division", Mathematics of Computation, Vol.44, No.170, pp.519-521, April, 1985) を利用することと、あるRNS基底で表現された整数を、基底拡張を用いて別の基底で表現すること、などが特徴として挙げられる。

【0009】ここで、基底拡張は、例えば、整数 x において、基底 $a = \{a_1, a_2, \dots, a_n\}$ での表現 $(x \bmod a_1, x \bmod a_2, \dots, x \bmod a_n)$ から基底 $a \cup \{b_1\} = \{a_1, a_2, \dots, a_n, b_1\}$ での表現 $(x \bmod a_1, x \bmod a_2, \dots, x \bmod a_n, x \bmod b_1)$ を生成する操作である。

【0010】この例では、基底のサイズを1つ増やしたが、この操作を繰り返して基底のサイズを n' 個 (n' は整数) 増やし、その基底の下でのRNS表現を作り出すことも基底拡張と呼ぶ。

【0011】続いて、以上のような剰余乗算方法について具体的に説明する。図4は川村により提案されたモンゴメリ乗算装置の構成を示す模式図である。このモンゴメリ乗算装置は、RNS表現でのモンゴメリ乗算を実行するものであり、本発明者並びに川村らによりCox-Roweアーキテクチャと命名された構成を備えている。係るCox-Roweアーキテクチャでは、互いに並列接続されてそれぞれ剰余乗算を行なう n 個のRowerユニットに対し、補正項を計算する1個のCoxユニットが接続されている。なお、 n は基底のサイズである。

【0012】ここで、Coxユニットは、基底拡張のためのものであり、ビット選択部11及び加算器12からなる。ビット選択部11は、RowerユニットのRAM21から入力された r ビットの整数 s_i の上位 q ビットを取り出して加算器12に与える。

【0013】加算器12は、その q ビットの整数を前回の加算結果に加算し、得られた加算結果の $q+1$ ビット目 (加算のキャリー1ビット) に位置する1ビットの数 k_i を各Rowerユニットに出力する。ここで r は基底 a の各要素 $a_1 \sim a_n$ のビット数であり、 q は r 未満の正整数である。

【0014】川村によれば、1,024ビットのべき乗剰余算のパラメータを $n=33$ 、 $r=32$ 、 $q=7$ と取れば良いことが分かる。従って、Coxユニットの処理は、7ビット程度の加算で済むので、高速に実行可能である。 n 個のRowerユニットは、RNSの基底を法とした剰余演算 $(fg + c + h) \bmod m$ を計算可能なものであり、RNS基底の下での剰余演算を行なう積和回路201 \sim 20 n 、RAM211 \sim 21 n 及びROM221 \sim

10

20

30

40

50

22_nからなる。ここで、mはRNS基底のある一要素であり、f、g、c、d（前回のh）はm未満の整数である。各積和回路20_i（但し1 ≤ i ≤ n）は、図5に示すように、(fg+c+d) mod mを計算するユニットであり、乗算器23、加算器24、スイッチ25、第1レジスタ26、第2レジスタ27、剰余演算器28及び第3レジスタ29を備えている。乗算器23は、入力f、gの積fgを算出して加算器24に出力する。加算器24は、その出力fgとスイッチ25からの入力c、及び第3レジスタ29内の前回の剰余演算結果dを加算し、得られた結果(fg+c+d)を剰余演算器28に出力する。なお、スイッチ25は、Coxユニットからの1ビットの入力k_iに基づいて、ROM22_iから第1レジスタ26に格納された入力cを加算器24に入力又は遮断するものである。

【0015】剰余演算器28は、ROM22_iから第2レジスタ27に格納された基底要素a_i又はb_iとしての法mを用い、加算器の出力結果fg+c+dを法mで割った剰余演算(fg+c+d) mod mを実行する。

【0016】ここで、ポッシュ又は川村のモンゴメリ乗算アルゴリズムによれば、RNSの基底としてm=2^r-μ（但し、μ≪2^r）の形式を適用可能なため、剰余演算器における剰余算z mod m、（但し、zは整数）を次のように加算と乗算で実行可能である。

【0017】

```
While (z < m) {
  z0 ← z mod 2r
  z1 ← z / 2r の商      (すなわち z = z1 2r + z0, z0 < 2r)
  z ← z1 μ + z0
}
```

なお、このアルゴリズムは次式に基づいている。

【0018】

$$\begin{aligned} z &= z_1 2^r + z_0 \\ &= z_1 (2^r - \mu + \mu) + z_0 \\ &= z_1 (2^r - \mu) + z_1 \mu + z_0 \\ &\equiv z_1 \mu + z_0 \pmod{m} \quad (m = 2^r - \mu \text{ のため}) \end{aligned}$$

係数z₀とz₁の計算は、剰余算(z₁μ+z₀) mod mがあるものの、z₀がzの下位rビットを抽出して得られ、z₁がzをrビット右シフトして得られるため、計算機で十分高速に実行可能である。

【0019】剰余演算器28は、このように剰余演算h=(fg+c+d) mod m=(z₁μ+z₀) mod mを実行する。ここで、実行中の剰余演算結果h'は、図示しないループ制御部により、法mと比較され、法mよりも大きい場合もしくは同じ場合(m ≤ h')、再度、入力側に戻されて前述した抽出・乗算・加算からなる剰余算が繰返され、法mよりも小さい場合(h' <

m)には剰余演算結果h（以下、剰余ともいう）として第3レジスタ29に出力される。なお、大小比較は、減算h'-mにより行われる。

【0020】第3レジスタ29内の剰余演算結果hは、積和回路20_iから出力されてRAM21_iに書込まれる一方、次回の積和回路20_iの演算時には前回の剰余演算結果dとして加算器23に与えられる。

【0021】

【発明が解決しようとする課題】しかしながら従来のモンゴメリ乗算装置及び方法では、ポッシュ又は川村の方法において、各積和回路20_iの剰余演算器28における剰余算mod mが加算や乗算と比較して処理量が多いので、Rowerユニットの演算を高速化、ひいてはCox-Rowerアーキテクチャによる演算を高速化するためのネックとなっている。

【0022】このため、剰余算mod mの更なる高速化が望まれている。

【0023】本発明は上記実情を考慮してなされたもので、剰余算の高速化により、少なくとも部分的な剰余演算を高速化でき、さらに全体的な演算速度の高速化を実現し得るモンゴメリ乗算装置及び方法を提供することを目的とする。

【0024】

【課題を解決するための手段】本発明の骨子は、Rowerユニットでの除算の条件を緩和し、剰余算を効率的に行なうことにあり、具体的には、Rowerユニットの積和回路における剰余算の終了条件を緩和することにある。

【0025】例えば、従来の剰余算h=z mod mの終了条件は、実行結果hがmよりも小さいとき(h < m)である。これに対し、本発明に係る剰余算h=z mod mの終了条件は、実行結果hが2^rよりも小さいとき(h < 2^r)である。ここで、rはmを2進表現したときのビット数である。すなわち、m < 2^rの関係がある。従って、本発明は、実行結果hが、法mよりも大きい値2^r未満に下がった時点で剰余算を終了するので、従来よりも剰余算を高速化することができる。

【0026】また、このように終了条件を緩和しても、後述する誤差の評価により演算の正確性を確認しているので、演算の信頼性を維持することができる。さて以上のような本発明の骨子に基づいて、具体的には以下のような手段が講じられる。請求項1に対応する発明は、正の整数Nと、N未満の正の整数x、yとを入力とし、剰余演算系の基底を{a₁, a₂, ..., a_i, ..., a_n}および{b₁, b₂, ..., b_i, ..., b_n} (1 ≤ i ≤ n, nは正の整数)としたとき、前記基底の要素の乗積として定義された整数B (= b₁b₂...b_n)を用いて、出力w=x y B⁻¹ mod Nを算出するためのモンゴメリ乗算装置であって、前記各基底の要素{a₁, a₂, ..., a_n}および{b₁, b₂, ..., b_n}と前記出力wの算出用の事前計算結果とが記憶される第1記憶

手段と、前記基底の拡張用の入力 ξ_i を受けたとき、前記入力 ξ_i の上位 q ビットを抽出すると共に、この今回の抽出値と前回の最上位ビットを除く加算結果とを加算し、得られた加算結果のうち、最上位ビットの値 k_i を出力する近似演算手段と、前記第1記憶手段の内容および前記近似演算手段の出力 k_i に基づいて、前記各基底の要素 $\{a_1, a_2, \dots, a_n\}$ および $\{b_1, b_2, \dots, b_n\}$ を法 m とする剰余算（但し、法 $m=2^r-\mu$ は基底の一要素、 r は $m \leq 2^r$ を満たす整数、 μ は 2^r 未満の非負整数）を実行する剰余演算手段と、前記剰余演算手段による剰余算の結果が記憶される第2記憶手段とを備え、前記剰余演算手段としては、前記第1記憶手段又は前記第2記憶手段からの入力 x, y の積 xy と前回の剰余算の結果 d と前記近似演算手段の出力 k_i に基づき入力される前記事前計算結果 c との和 $(xy+d+c)$ からなる入力 z （但し、 z は正の整数）を受けると共に、前記入力 z の法 m による剰余算 $h=z \bmod m$ を実行する剰余算部と、前記剰余算部による実行結果 h' と前記法 m の上限値 2^r とを比較し、前記実行結果 h' が 2^r 以上のときには当該実行結果 h' を前記入力 z として前記剰余算部に戻し、前記実行結果 h' が 2^r 未満のときには当該実行結果 h' を前記剰余算の結果 h として前記第2記憶手段に出力するループ制御部と、を備えたモンゴメリ乗算装置である。

【0027】また、請求項2に対応する発明は、請求項1に対応するモンゴメリ乗算装置において、前記剰余算部としては、前記入力 z の法 m による剰余算 $h=z \bmod m$ を実行するとき、前記入力 z の下位 r ビットの値 z_0 と、前記入力 z から前記下位の値 z_0 を除いた上位ビットの値 z_1 とを夫々抽出するビット選択部と、前記ビット選択部により抽出された上位ビットの値 z_1 と前記法 m に関する入力 μ とを受けたとき、両者の積 $z_1\mu$ を算出する乗算部と、前記乗算部により得られた積 $z_1\mu$ と前記ビット選択部により抽出された下位 r ビットの値 z_0 との和 $(z_1\mu+z_0)$ を算出する加算部とを備え、前記剰余算部による実行結果 h' は、前記加算部により得られた和 $(z_1\mu+z_0)$ であるモンゴメリ乗算装置である。

【0028】さらに、請求項3に対応する発明は、正の整数 N と、 N 未満の正の整数 x, y とを入力とし、剰余演算系の基底を $\{a_1, a_2, \dots, a_i, \dots, a_n\}$ および $\{b_1, b_2, \dots, b_i, \dots, b_n\}$ （ $1 \leq i \leq n$ 、 n は正の整数）としたとき、前記基底の要素の乗積として定義された整数 $B(=b_1b_2 \dots b_n)$ を用いて、出力 $w=xyB^{-1} \bmod N$ を算出するためのモンゴメリ乗算方法であって、前記各基底の要素 $\{a_1, a_2, \dots, a_n\}$ および $\{b_1, b_2, \dots, b_n\}$ と前記出力 w の算出用の事前計算結果とが記憶される第1記憶工程と、前記基底の拡張用の入力 ξ_i を受けたとき、前記入力 ξ_i の上位 q ビットを抽出すると共に、この今回

の抽出値と前回の最上位ビットを除く加算結果とを加算し、得られた加算結果のうち、最上位ビットの値 k_i を出力する近似演算工程と、前記第1記憶工程の記憶内容および前記近似演算工程の出力 k_i に基づいて、前記各基底の要素 $\{a_1, a_2, \dots, a_n\}$ および $\{b_1, b_2, \dots, b_n\}$ を法 m とする剰余算（但し、法 $m=2^r-\mu$ は基底の一要素、 r は $m \leq 2^r$ を満たす整数、 μ は 2^r 未満の非負整数）を実行する剰余演算工程と、前記剰余演算工程による剰余算の結果が記憶される第2記憶工程とを含んでおり、前記剰余演算工程としては、前記第1記憶工程の記憶内容又は前記第2記憶工程の記憶内容からの入力 x, y の積 xy と前回の剰余算の結果 d と前記近似演算工程の出力 k_i に基づき入力される前記事前計算結果 c との和 $(xy+d+c)$ からなる入力 z

（但し、 z は正の整数）を受けると共に、前記入力 z の法 m による剰余算 $h=z \bmod m$ を実行する剰余算工程と、前記剰余算工程による実行結果 h' と前記法 m の上限値 2^r とを比較し、前記実行結果 h' が 2^r 以上のときには当該実行結果 h' を前記入力 z として前記剰余算工程に戻し、前記実行結果 h' が 2^r 未満のときには当該実行結果 h' を前記剰余算の結果 h として前記第2記憶工程に出力するループ制御工程と、を含んでいるモンゴメリ乗算方法である。

【0029】また、請求項4に対応する発明は、請求項3に対応するモンゴメリ乗算方法において、前記剰余算工程としては、前記入力 z の法 m による剰余算 $h=z \bmod m$ を実行するとき、前記入力 z の下位 r ビットの値 z_0 と、前記入力 z から前記下位の値 z_0 を除いた上位ビットの値 z_1 とを夫々抽出するビット選択工程と、前記ビット選択工程により抽出された上位ビットの値 z_1 と前記法 m に関する入力 μ とを受けたとき、両者の積 $z_1\mu$ を算出する乗算工程と、前記乗算工程により得られた積 $z_1\mu$ と前記ビット選択工程により抽出された下位 r ビットの値 z_0 との和 $(z_1\mu+z_0)$ を算出する加算工程とを含んでおり、前記剰余算工程による実行結果 h' は、前記加算工程により得られた和 $(z_1\mu+z_0)$ であるモンゴメリ乗算方法である。

【0030】（作用）従って、請求項1, 3に対応する発明は以上のような手段を講じたことにより、剰余演算手段としては、剰余算部が、第1記憶手段又は第2記憶手段からの入力 x, y の積 xy と前回の剰余算の結果 d と近似演算手段の出力 k_i に基づき入力される事前計算結果 c との和 $(xy+d+c)$ からなる入力 z （但し、 z は正の整数）を受けると共に、入力 z の法 m による剰余算 $h=z \bmod m$ を実行し、ループ制御部が、剰余算部による実行結果 h' と法 m の上限値 2^r とを比較し、実行結果 h' が 2^r 以上のときには当該実行結果 h' を入力 z として剰余算部に戻し、実行結果 h' が 2^r 未満のときには当該実行結果 h' を剰余算の結果 h として第2記憶手段に出力する。

【0031】このように、法 m での剰余算であるにも関わらず、剰余算の実行結果 h' を法 m よりも大きい値 2^r と比較するように剰余算の終了条件を緩和したので、剰余算の高速化により、少なくとも部分的な剰余演算を高速化でき、さらに全体的な演算速度の高速化を実現させることができる。

【0032】また、剰余算の終了条件を緩和したが、後述する誤差の評価により演算の正確性を確認しているので、演算結果の信頼性を維持することができる。

【0033】また、請求項2、4に対応する発明は、剰余算部としては、ビット選択部が、入力 z の法 m による剰余算 $h = z \bmod m$ を実行するとき、入力 z の下位 r ビットの値 z_0 と、入力 z から下位の値 z_0 を除いた上位ビットの値 z_1 とを夫々抽出し、乗算部が、ビット選択部により抽出された上位ビットの値 z_1 と法 m に関する入力 μ とを受けたとき、両者の積 $z_1\mu$ を算出し、加算部が、乗算部により得られた積 $z_1\mu$ とビット選択部により抽出された下位 r ビットの値 z_0 との和 $(z_1\mu + z_0)$ を算出し、剰余算部による実行結果 h' を、加算部により得られた和 $(z_1\mu + z_0)$ としたので、剰余算をビットの抽出・加算・乗算などの簡単な処理で実行でき、請求項1、3に対応する作用を容易且つ確実に奏することができる。

【0034】

【発明の実施の形態】以下、本発明の一実施形態について図面を参照して説明する。始めに、本発明のモンゴメリ乗算装置及び方法に適用されるモンゴメリ乗算アルゴリズムについて説明する。まず、RNS表現におけるモンゴメリ乗算アルゴリズムを説明する。ここで、モンゴメリ乗算の法を N とし、入力を $x, y < 2N$ とする。出力は、 $w = xyB^{-1} \bmod N$ 、又は $w = xyB^{-1} \bmod N+N$ である。係るモンゴメリ乗算アルゴリズムは、次のような7つのステップST1～ST7で表せる。

ST1. $\langle s \rangle_a \leftarrow \langle x \rangle_a \langle y \rangle_a, \langle s \rangle_b \leftarrow \langle x \rangle_b$

$$\beta = \sum_{i=1}^n \beta_i x (A_i^{-1} \bmod a_i) \times A_i \bmod A \quad \dots (1)$$

【0040】但し、 $A_i = A/a_i$ であり、 A_i^{-1} は法 a_i における A_i の乗法逆元である。このとき(1)式の $\bmod A$ を次の(2)式のように減算でも表せる。 ※

$$\beta = \sum_{i=1}^n \beta_i x (A_i^{-1} \bmod a_i) \times A_i - kA \quad \dots (2)$$

【0042】(2)式のように、 $\bmod A$ を減算で表すとき、 $-A$ の倍数となる非負整数 k がただ一つ存在する。この $-A$ の倍数 k の計算が、ボッシュや川村のRNS表現でのモンゴメリ乗算の処理速度を決めるポイントとなっている。

* $\langle y \rangle_b$

ST2. $\langle t \rangle_b \leftarrow \langle s \rangle_b \langle (-N)^{-1} \rangle_b$

ST3. $\langle t \rangle_b$ から基底拡張により $\langle t \rangle_{a \cup b}$ を求める。

ST4. $\langle u \rangle_a \leftarrow \langle t \rangle_a \langle N \rangle_a$

ST5. $\langle v \rangle_a \leftarrow \langle s \rangle_a + \langle u \rangle_a$

ST6. $\langle w \rangle_a \leftarrow \langle v \rangle_a \langle B^{-1} \rangle_a$

ST7. $\langle w \rangle_a$ から基底拡張により $\langle w \rangle_{a \cup b}$ を求める。

【0035】ここで、 $\langle s \rangle_a$ は、基底 a の下での s のRNS表現を表す。その他の $\langle t \rangle_b$ なども同様である。例えば $\langle s \rangle_a$ は、 $A = a_1 a_2 \dots a_n$ を法とする剰余環の元 s を基底 a の各要素 $a_1 \sim a_n$ で割った余りの組 (s_1, s_2, \dots, s_n) を表す。

【0036】基底 $b = \{b_1, b_2, \dots, b_n\}$ は基底 a とは別のRNS基底であり、 $B = b_1 b_2 \dots b_n$ としたとき、 A と B の最大公約数 $\gcd(A, B) = 1$ である基底である。ステップST1～ST7の正確な計算には、少なくとも $N < A, N < B$ という条件が必要である。この条件から x や y は、基底 a のみ、あるいは基底 b のみで一意的に表現できるので、入力 x を $\langle x \rangle_a, \langle x \rangle_b$ のペアで表すことは冗長である。しかし、入力 x, y は、両者の積 s のとり範囲が $0 \leq s < N^2$ であるため、基底 a, b の合併集合 $a \cup b$ を基底 $a \cup b$ として始めて正しく表現される。

【0037】次に、以上のようなモンゴメリ乗算アルゴリズムのうち、ステップST3, ST7の基底拡張について説明する。いま β を A 未満の非負整数とし($0 \leq \beta < A$)、非負整数 β のRNS表現を $\langle \beta \rangle_a = (\beta_1, \beta_2, \dots, \beta_n)$ とする。ここで、 $\langle \beta \rangle_a$ から基底拡張により $\langle \beta \rangle_{a \cup b}$ を求めるとする。

【0038】このとき、周知の中国剰余定理から次の(1)式が成り立つ。

【0039】

【数1】

※【0041】

【数2】

【0043】次に、倍数 k を算出する際に、効率の良い川村のアルゴリズムを説明する。(2)式の両辺を A で割ると、次の(3)式になる。

【0044】

【数3】

$$\beta/A = \sum_{i=1}^n (\beta_i \times A_i^{-1} \bmod a_i) / a_i - k \quad \dots (3)$$

(3) 式を変形すると次の(4)式のようになる。

$$k = \sum_{i=1}^n (\beta_i \times A_i^{-1} \bmod a_i) / a_i - \beta/A \quad \dots (4)$$

(4) 式から $0 \leq \beta/A < 1$ を用いて次の(5)式の関係を表せる。

$$k \leq \sum_{i=1}^n (\beta_i \times A_i^{-1} \bmod a_i) / a_i < k+1 \quad \dots (5)$$

【0045】ここで、小数部の切捨て操作を記号 $[]$ で表すと、(5)式に基づき、次の(6)式のように倍数 k を表すことができる。

$$k = \left[\sum_{i=1}^n (\beta_i \times A_i^{-1} \bmod a_i) / a_i \right] \quad \dots (6)$$

【0047】この(6)式は、除算を含むため、川村は次の(7)式のように近似計算をして倍数 k' を求めている。

【0048】

【数5】

$$k' = \left[\sum_{i=1}^n \text{trunc}(\xi_i) / 2^r \right] \quad \dots (7)$$

【0049】ここで、 $\xi_i = \beta_i \times A_i^{-1} \bmod a_i$ である。 $\text{trunc}()$ は、 r ビットの変数 ξ_i のうち、上位 q ビットを維持して上位 q ビット目より下位の $(r-q)$ ビットを全て0にする切捨て関数である。 r は基底要素のビット数であり、 q は r 未満の正整数である。

【0050】実際に(7)式は、ハードウェアで計算されるとき、以下のように逐次的に実行される。

$$\varepsilon_a = \max \{ (2^r - a_i) / 2^r \} \quad \dots (8)$$

$$\delta_a = \max \{ (\xi_i - \text{trunc}(\xi_i)) / a_i \} \quad \dots (9)$$

なお、 $\max \{ \}$ は i を $1, 2, \dots, n$ と動かしたときの最大値を表す。川村によれば、これら評価変数 ε_a 、 δ_a を使い、入力 β が次の(10)式の範囲にあるとき、近似計算の(7)式は正しい k の値を与える。

【0053】

$$n(\varepsilon_a + \delta_a)A \leq \beta < A \quad \dots (10)$$

また、川村によれば、入力 β が次の(11)式の範囲内のとき、近似計算の(7)式は正しい k 又は $k-1$ の値を与える。

* 【0046】

【数4】

*

```

※for (i=0; i<=n; i++) {
20  σi = σi-1 + trunc(ξi) / 2r
    ki = [σi]
    σi = σi - ki
}

```

ここで、 $\sigma_0 = \alpha$ 、 $0 \leq \alpha < 1$ を初期値とする。 α は誤差の補正のために導入されている。また、規則正しい処理を行なう観点から、ビット数 r は全ての基底要素で共通とするのが望ましく、ここではそのように仮定する。

【0051】(7)式で近似計算した倍数 k' は、

(6)式で得られる正確な倍数 k に対して近似誤差が生じる。そこで、(7)式の近似誤差の評価のため、次の

(8)～(9)式に示すように、(7)式の分母の評価変数 ε_a と、分子の評価変数 δ_a とが導入される。

【0052】

★ 【0054】

$$0 \leq \beta < n(\varepsilon_a + \delta_a)A \quad \dots (11)$$

また、(7)式に基づいて、近似値 k' を計算し、次の(12)式を各 $i=1, 2, \dots, n$ について実行することで、基底 a から基底 b の表現を作り出すことができる。但しこの(12)式の場合は誤差が生じる可能性がある。

【0055】

【数6】

★

$$\beta \bmod b_i = \sum_{i=1}^n (\xi_i \times A_i \bmod b_i + k_i (b_i - A_i \bmod b_i)) \bmod b_i \quad \dots (12)$$

【0056】以上の説明では、基底 a から基底 b への拡張方法を述べたが、記号 a と記号 b との交換により、基

底 b から基底 a への基底拡張方法、および評価変数 ε_b 、 δ_b が得られる。

【0057】さらに、川村によれば、この近似値 k' を使った基底拡張アルゴリズム(12)式を用いて、ステップST1~ST7のRNS表現でのモンゴメリ乗算アルゴリズムを実行した場合、次の5つの条件(13)~(17)式を満たすときには、 $w = xyB^{-1} \bmod N$ 、 $w < 2N$ なる $\langle w \rangle_a \cup b$ を出力することが示されている。ここで、 $\Delta = n(\varepsilon + \delta)$ 、 $\varepsilon = \max\{\varepsilon_a, \varepsilon_b\}$ 、 $\delta = \max\{\delta_a, \delta_b\}$ である。

【0058】 $\gcd(A, B) = 1 \quad \dots (13)$
 $\gcd(B, N) = 1 \quad \dots (14)$
 $0 \leq \Delta \leq \alpha < 1 \quad \dots (15)$
 $4N / (1 - \Delta) \leq B \quad \dots (16)$
 $2N / (1 - \alpha) \leq A \quad \dots (17)$

以上が本発明に適用されるモンゴメリ乗算アルゴリズムである。次に、以上のようなモンゴメリ乗算アルゴリズムを適用対象とする本発明の一実施形態について図面を参照しながら説明する。

【0059】図1は本発明の一実施形態に係るモンゴメリ乗算装置の構成を示す模式図であり、図2はこのモンゴメリ乗算装置内の積和回路の構成を示す模式図であり、図3はこの積和回路内の除算回路の構成を示す模式図であって、前述した図面と同一部分には同一符号を付してその詳しい説明を省略し、ここでは異なる部分について主に述べる。

【0060】すなわち、本実施形態は、前述したCox-Rowerユニットの高速化を図るものであり、具体的には図1~図3に示すように、前述した剰余演算器28を有する各積和回路201~20nに代えて、除算回路40を有する積和回路301~30nを備えている。

【0061】ここで、各積和回路30i(但し $1 \leq i \leq n$)は、図2に示すように、 $(fg + c + d) \bmod m$ を計算するユニットであり、前述した剰余演算器28に代えて、除算回路40を備えている。なお、mは、前述同様にRNS基底のある一要素である。但し、f, g, c, dは、従来のm未満の整数とは異なり、 2^r 以下の正整数である。

【0062】除算回路(剰余演算器)40は、加算器24の出力結果 $fg + c + d$ を法mで割った剰余hを計算するものであり、ハードウェア又はソフトウェアのいずれでもよいが、図3に示すように、ビット選択部41、乗算器42、加算器43及びループ制御部44を備えている。

【0063】ビット選択部41は、入力された整数zを下位rビットの値 z_0 と、入力された整数zから下位rビットを除いた上位ビットの値 z_1 とを夫々抽出する機能を持っている。ここで、値 z_0 、 z_1 は、夫々次式を満たすものとなる。

【0064】 $z_0 = z \bmod m$
 $z_1 = \lfloor z / 2^r \rfloor$

乗算器42は、ビット選択部41により抽出された上位

ビットの値 z_1 と、法mを $m = 2^r - \mu$ と表したときの μ との積 $z_1 \mu$ を計算し、得られた積 $z_1 \mu$ を加算器43に出力する機能をもっている。

【0065】加算器43は、ビット選択部41で選択された z_0 と乗算器42の出力 $z_1 \mu$ を加算する。

【0066】ループ制御部44は、加算器43の出力結果 $z_1 \mu + z_0$ を法mの上限値 2^r と比較し、大きい場合もしくは同じ場合($2^r \leq z_1 \mu + z_0 (=h')$)は $z_1 \mu + z_0$ を改めてzとしてビット選択部41に投入して処理を繰返し、小さい場合は $z_1 \mu + z_0$ を剰余算結果hとして第3レジスタ29を介してRAM21jに出力するように制御する機能をもっている。すなわち、除算回路40の出力hは、 $h = z \bmod m$ 、 $h < 2^r$ を満たすものである。

【0067】次に、以上のように構成されたモンゴメリ乗算装置によるモンゴメリ乗算方法について説明する。

【0068】いま、本実施形態装置は、入力x, y, 法N、基底a, bを準備し、ROM211~21nに各基底の要素 $\{a_1, a_2, \dots, a_n\}$ および $\{b_1, b_2, \dots, b_n\}$ と出力wの算出用の事前計算結果とを記憶させた後、ステップST1~ST7のモンゴメリ乗算アルゴリズムを実行する。また、ステップST3, ST7の基底拡張は、川村のアルゴリズム(12)式に従って実行される。このとき、Coxユニットとしては、基底の拡張用の入力 ξ_i を受けると、入力 ξ_i の上位qビットを抽出すると共に、この今回の抽出値と前回の最上位ビットを除く加算結果とを加算し、得られた加算結果のうち、最上位ビットの値 k_i を各Rowerユニットの積和回路301~30nに出力する。

【0069】さて、ここでモンゴメリ乗算アルゴリズムST1~ST7内の各基底成分の計算について詳述する。例えば、ST1の計算 $\langle s \rangle_a \leftarrow \langle x \rangle_a \cdot \langle y \rangle_a$ の第一成分 $x_1 \cdot y_1 \bmod a_1$ を考える。

【0070】この計算は、図1中、第1のRowerユニットで行なわれ、剰余乗算が第1の積和回路301により実行される。具体的には図2に示すように、まず積和回路301に入力された x_1 、 y_1 が乗算器23に入力される。乗算器23はその積 $x_1 \cdot y_1$ を計算して加算器24に出力する。

【0071】この計算では、スイッチ25が開いているので入力cが無く、前回の演算結果dも0である。このため、積 $x_1 \cdot y_1$ と $d = 0$ とが加算器24に入力される。

【0072】加算器24はこの和を計算して除算回路40に出力する。この結果 $x_1 \cdot y_1$ と、基底 a_1 を $a_1 = 2^r - \mu_1$ としたときの μ_1 とが除算回路40に入力される。

【0073】ここで、除算回路40は、 $h = x_1 \cdot y_1 \bmod a_1$ (但し $h < 2^r$)を出力する。なお、この除算回路40は、法 a_1 の除算を実行するものの、従

来とは異なり、大小比較の対象を 2^r として除算結果 h を出力する。なお、除算回路40は、大小比較の対象を 2^r としたので、図3に示した構成に限らず、剰余算を高速化することができる。但し、ここでは図3に示した構成を例に挙げて述べる。

【0074】まず、整数 $z = x_1 \cdot y_1$ が除算回路40に入力される。

【0075】除算回路40においては、ビット選択部41が、下位 r ビットの値 $z_0 = z \bmod 2^r$ と、残りの上位ビットの値 $z_1 = \lfloor z / 2^r \rfloor$ とを抽出し、上位ビットの値 z_1 と入力 μ_1 とが乗算器42に入力される。

【0076】乗算器42は積 $z_1 \mu_1$ を計算して加算器43に出力する。この出力 $z_1 \mu_1$ と、ビット選択部41で抽出された下位ビットの値 z_0 とが加算器43に入力される。

【0077】加算器43は、これらの和 $h = z_1 \mu_1 + z_0$ を計算して出力する。

【0078】ループ制御部44は、加算器43からの出力 h' を法 m の上限値 2^r と大小比較し、出力 h' が 2^r 以上のとき、出力 h' を改めてビット選択部の入力 z として処理を繰り返す。

【0079】また、ループ制御部44は、大小比較の結果、出力 h が 2^r 未満のときにはループ終了と判断し、出力 h' を剰余 h として出力する。なお、本来、法 a_1 の剰余算を行なうものであるが、出力 h が法 a_1 より大でもよい。

【0080】このとき $h = x_1 \cdot y_1 \bmod a_1$ 、 $h < 2^r$ を満たす。 h は、積和回路301から出力され、RAM211に格納される。

【0081】この部分以外にもRowerユニットでの剰余演算が必要なときは同様の処理を行なう。ここで、本発明の除算回路40による剰余算の高速化について補足的に説明する。従来は、図示しないループ制御部にて、剰余

$$n(\varepsilon \cdot 2^\tau / (2^\tau - 1) + \delta) A \leq \beta < 1 \quad \dots (10a)$$

また、 β が次の(11a)式の範囲内にあれば、(7)式の近似計算は正しい k 又は $k-1$ を与える。 ※

$$0 \leq \beta < n(\varepsilon \cdot 2^\tau / (2^\tau - 1) + \delta) A \quad \dots (11a)$$

これら(10a)、(11a)式は、それぞれ前述した(10)、(11)式に対応した条件であり、考えている状況も(10)、(11)式の状況と同じとする。また、モンゴメリ乗算も前述した5つの条件(13)～(17)式のうち、補正項 α の範囲を定めた(15)式★

$$0 \leq n(\varepsilon \cdot 2^\tau / (2^\tau - 1) + \delta) A \leq \alpha < 1 \quad \dots (15a)$$

実用上、 N が1,024ビットの場合を考えると、 $n=3$ 、 $\alpha=0.5$ と取れば良く、このとき上の τ は、 $\tau \geq 2$ 程度と取れることが確認できる。

【0091】このとき $2^\tau / (2^\tau - 1)$ は1に十分近く、(10a)、(11a)、(15a)式の条件は、それぞれ(10)、(11)、(15)式と対応してい

* 余演算結果 h' と、基底の一要素(例、 a_1)とを大小比較するため、減算 $h' - a_1$ を行ない、その減算結果が負になるか否かを確認することが必要である。

【0082】一方、本発明は、加算器43からの出力 h' と、基底要素のビット数 r で示せる上限値 2^r とを大小比較するため、出力 h' が $(r+1)$ ビット以上であるか否かを確認すればよいので、従来の減算よりも、効率的かつ高速にループの制御を実行できる。

【0083】本発明は、このようなループ制御自体の高速化に加え、ループ回数自体も減少していることにより高速化を実現している。例えば、本願発明でのループ回数は、出力 h が上限値 2^r 未満($h < 2^r$)に減少するまでの繰返し回数である。一方、従来のループ回数は、出力 d が法 a_1 未満($d < a_1$)に減少するまでの繰返し回数である。ここで、基底の要素 a_1 はそのビット数 r における上限値 2^r よりも小さい($a_1 < 2^r$)。

【0084】従って、一般に、出力 h が上限値 2^r 未満に低減するまでのループ回数の方が、出力 d が法 a_1 未満に低減するまでのループ回数よりも少ない。すなわち、本願発明のループ回数の方が従来のループ回数よりも通常少ない。

【0085】すなわち、本発明は、このようにループ回数が少ないことにもより、剰余算の高速化を実現することができる。(誤差の評価)本発明のモンゴメリ乗算装置及び方法を用いてステップST1～ST7のモンゴメリ乗算アルゴリズムを実行するときの誤差の評価は次の各式で与えられる。

【0086】いま、分母の評価関数 ε を、 $\varepsilon < 2^{-\tau}$ (但し $\tau > 0$)の範囲とする。ここで、 τ は、 ε の範囲を規定するために導入した正整数である。

【0087】このとき、 β が次の(10a)式の範囲内にあれば、(7)式の近似計算は正しい k の値を与える。

【0088】

※【0089】

★を次の(15a)式と置換することで、 $w = xyB^{-1} \bmod N$ 、または $w = xyB^{-1} \bmod N+N$ を出力する。

【0090】

るが、これらとほぼ同じ条件となっている。

【0092】すなわち、本実施形態を実行した場合であっても、誤差を正確に評価できるので、本実施形態の装置及び方法の信頼性を確保することができる。上述したように本実施形態によれば、法 m での剰余算であるにも関わらず、剰余算の実行結果 h' を法 m よりも大きい値

2rと比較するように剰余算の終了条件を緩和したので、2rという比較の容易な数の採用によるループ制御自体の高速化と、2rという大きい数の採用によるループ回数の低減化とによる剰余算の高速化により、少なくとも部分的な剰余演算を高速化でき、さらに全体的な演算速度の高速化を実現させることができる。

【0093】また、剰余算の終了条件を緩和したが、(10a)、(11a)、(15a)式からなる誤差の評価により演算の正確性を確認しているので、演算結果の信頼性を維持することができる。

【0094】また、除算回路40が、ビット選択部41、乗算部42、加算部43を有し、実行結果h'を、加算部により得られた和($z_1\mu + z_0$)とする場合、剰余算をビットの抽出・加算・乗算などの簡単な処理で実行できるので、前述した効果を容易且つ確実に奏することができる。

【0095】また、本実施形態に係るCox-Rowerアーキテクチャは、例えばチップを搭載したボード等により実現でき、このボードをサーバ装置にさし込むことにより、サーバ上に実現させることもできる。

【0096】なお、上記実施形態に記載した手法は、コンピュータに実行させることのできるプログラムとして、磁気ディスク（フロッピー（登録商標）ディスク、ハードディスクなど）、光ディスク（CD-ROM、DVDなど）、光磁気ディスク（MO）、半導体メモリなどの記憶媒体に格納して頒布することもできる。

【0097】なお、本願発明は、上記各実施形態に限定されるものでなく、実施段階ではその要旨を逸脱しない範囲で種々に変形することが可能である。また、各実施形態は可能な限り適宜組み合わせで実施してもよく、その場合、組み合わせられた効果が得られる。さらに、上記各実施形態には種々の段階の発明が含まれており、開示

される複数の構成要件における適宜な組み合わせにより種々の発明が抽出され得る。例えば実施形態に示される全構成要件から幾つかの構成要件が省略されることで発明が抽出された場合には、その抽出された発明を実施する場合には省略部分が周知慣用技術で適宜補われるものである。

【0098】その他、本発明はその要旨を逸脱しない範囲で種々変形して実施できる。

【0099】

- 10 【発明の効果】以上説明したように本発明によれば、剰余算の高速化により、少なくとも部分的な剰余演算を高速化でき、さらに全体的な演算速度の高速化を実現し得るモンゴメリ乗算装置及び方法を提供できる。

【図面の簡単な説明】

【図1】本発明の一実施形態に係るモンゴメリ乗算装置の構成を示す模式図

【図2】同実施形態における積和回路の構成を示す模式図

- 20 【図3】同実施形態における除算回路の構成を示す模式図

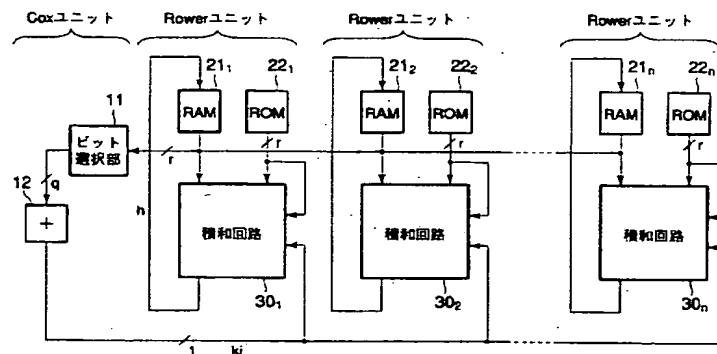
【図4】従来のモンゴメリ乗算装置の構成を示す模式図

【図5】従来の積和回路の構成を示す模式図

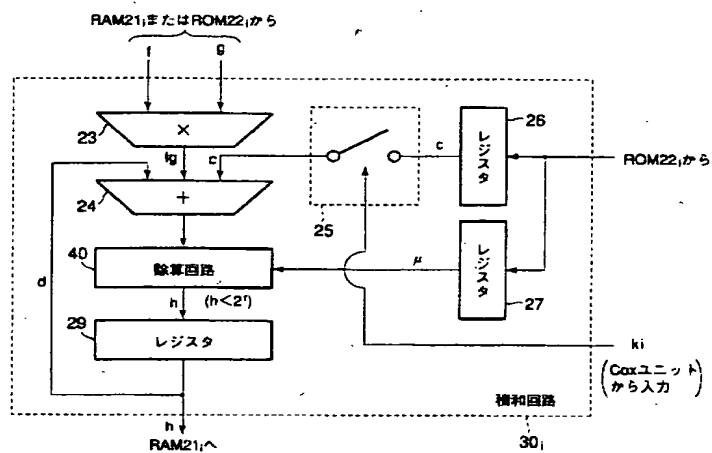
【符号の説明】

- 11, 41…ビット選択部
12, 24, 43…加算器
23, 42…乗算器
25…スイッチ
26, 27, 29…レジスタ
301~30n…積和回路
40…除算回路
44…ループ制御部

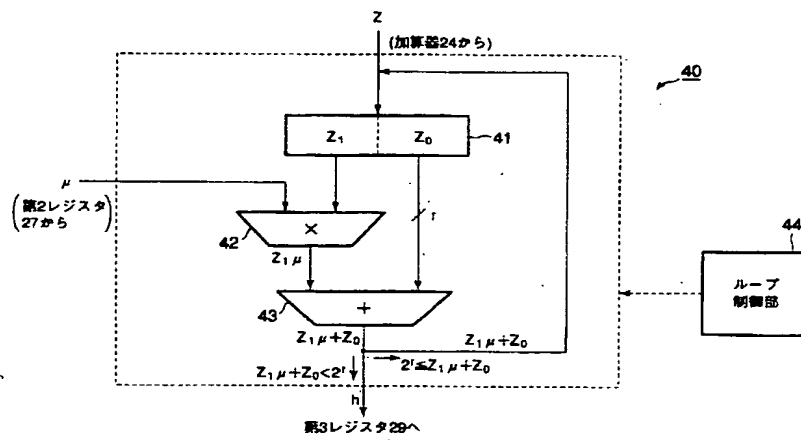
【図1】



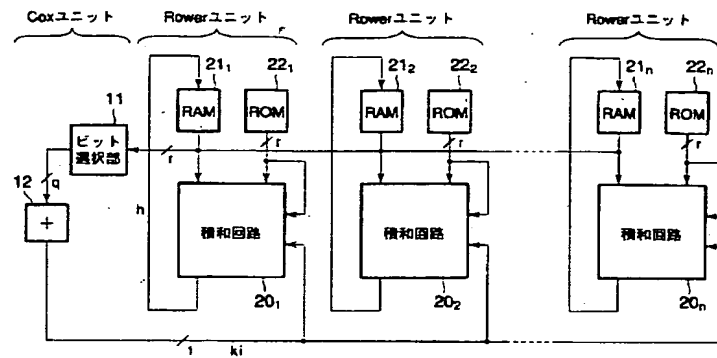
【図2】



【図3】



【図4】



【図5】

